Delta Plugin

Bacula Systems Documentation

Contents

Contents

Important: Delta Plugin is not under active development anymore. The solution has been enhanced, and moved to more recent plugins: ged and aligned-volumes.

See also:

- ged
- Aligned Volumes

1 Overview

This white paper presents various techniques and strategies to backup special files such as Outlook PST, Database datafile, VirtualBox/VmWare images using block level analysis with **Bacula Enterprise**.

1.1 Scope

This paper will present solutions for **Bacula Enterprise** 6.0 and later, which are not applicable to prior versions.

2 Presentation

The Delta plugin is designed to analyse file differences at the block level and generate binary patches for the backup. It should permit significant space reduction on files that have few modifications between two backups. This plugin is available on all platforms such as Linux and Windows 32/64bit.

3 Taking Advantage of Delta Backups

Special files such as :

- Outlook PST
- Unix Mbox
- Database
- VMware/VirtualBox/KVM images

will generally benefit most from this technology, other files like :

- Videos
- Photos
- Windows/Linux binaries
- Compressed files
- Encrypted files

probably will not have significant compression ratios using the Delta plugin. The main reason is that most of the time, a single byte modification in an editor will modify all blocks, so the binary patch may be larger than the original file.

4 Using Delta Plugin

4.1 Configuration

As with all Bacula plugins, you must to specify the `Plugin Directory` directive in the `FileDaemon` resource of the `bacula-fd.conf` file.

```
FileDaemon {
  Name = test-fd
  ...
  Plugin Directory = /opt/bacula/plugins
}
```

The `delta` plugin uses Accurate mode information to handle incremental and differential backups, thus you must enable the Accurate option in your Job resource.


```
Job {
 Name = "Outlook_PST"
 Client = laptop1-fd
 FileSet = FS_PST
 Accurate = yes
 ...
}
```

To activate the delta plugin on a particular pattern of files such as all **.pst** files, use the following directives:

```
FileSet {
 Name = FS_PST
 Include {
   Options {
     Signature = MD5
     Plugin = delta
     WildFile = "*.pst"
   }
   File = c:/Outlook
 }
}
```

Please note that in the above case, the `Plugin` is specified inside the `Options` block rather than at the same level as the `File` directive.

The Delta plugin also accepts the following parameters:

Table 1: Delta Plugin Options

Option	Comment	Example
min_size=x	Don't use the Delta plugin on files smaller than `x`	min_size=4k

```
FileSet {
 Name = FS_PST
 Include {
   Options {
     Signature = MD5
     Plugin = "delta: min_size=512k"
   }
   File = /home/bacula
 }
}
```

In this example, we split the FileSet to use the Delta plugin on PST files, and the standard compression for other files.

```
FileSet {
 Name = FS_PST
 Include {
   Options {
     Signature = MD5
     Plugin = delta
     WildFile = "*.pst"
   }
   File = c:/Outlook
```

(continues on next page)

(continued from previous page)

```
 }
 Include {
    Options {
      Signature = MD5
      Compression = LZO
    }
    File = "C:/User"
  }
}
```

In this example, only the MYI/MYD files are selected.

```
FileSet {
  Name = FS_MYSQL
  Include {
    Options {
      Signature = MD5
      Plugin = "delta"
      Wildfile = "*.MYI"
      Wildfile = "*.MYD"
    }
    Options {
      Plugin = "delta"
      Exclude = yes
      Wildfile = "*"
    }
    File = /var/lib/mysql
  }
}
```

4.2 Requirements

To analyse differences between file versions, the `delta` plugin needs to store signatures in the working directory for each file backed up. Those signatures require some space on your client machine, but don't need to be backed up. They are stored in the `working/delta/` directory. At this time, Bacula will not cleanup the delta working directory, however, you may want to remove entries after some time. You can use the following command for this purpose. Note that you may need to adapt the `60` days parameter to your particular configuration to ensure that you do not remove signatures that are still needed:

```
% find /opt/bacula/working/delta/ -type f -mtime +60 -exec rm -f {} \;
```

The `delta` plugin also needs to store information specific to backup jobs. This information is kept in the `working/delta.history` file.

During restore, Bacula will restore the first backup of the file in a temporary location, then it will apply patches to the temporary file for each incremental/differential backup that was made, finally Bacula will move the temporary file into the right place. This means that you will need working storage to perform the restore. For example, to restore a 2G file, you will need at least 4GB of temporary space.

4.3 Development Direction

We are considering implementing the following items in a future version:

- Store signatures in a more efficient way
- Cleanup signatures automatically
- Exclude some file type automatically (jpg, mpg, avi, mkv)
- Apply Delta patches "inplace" rather than using a temporary file

4.4 Installation

The `delta` plugin is available as a Bacula Enterprise package for all supported platforms.

Note: More than with traditional Incremental backups, you need to **ensure** that all dependent jobs are available for restore. If even one of your Incremental jobs is missing at restore time, Bacula will not be able to restore your file correctly. In otherwords, if even a single delta is missing, the file cannot be reconstructed. As with normal backup, using the Differential level permits reducing the number of Jobs that are required for restore.

4.5 Current Limitations

- On Windows, file attributes such as ACLs are not handled by the Delta plugin.
- The Delta plugin is not compatible with Base job level backups.
- The Delta plugin is not compatible with the `sparse` FileSet option.
- The Delta plugin is not compatible with other plugins in the same FileSet

